Joint source-channel coding/decoding of 3D-ESCOT bitstreams

Manel Abid, Michel Kieffer, Beatrice Pesquet-Popescu

Joint source-channel coding/decoding of 3D-ESCOT bitstreams

M. Abid[1], and M. Kieffer[1,2], and B. Pesquet-Popescu[1]

[1] *Institut Télécom, Télécom ParisTech and CNRS LTCI*
Signal and Image Processing Department,
46 rue Barrault, 75634 Paris Cedex 13, France

[1] *on sabbatical leave from L2S - CNRS - SUPELEC - Univ Paris-Sud*
3 rue Joliot-Curie, 91192 Gif-sur-Yvette, France.

***Abstract*—Joint source-channel decoding (JSCD) exploits residual redundancy in compressed bitstreams to improve the robustness to transmission errors of multimedia coding schemes. This paper proposes an architecture to introduce some additional side information in compressed streams to help JSCD. This architecture exploits a reference decoder already present or introduced at the encoder side. An application to the robust decoding of 3D-ESCOT encoded bitstreams generated within the Vidwav video coder is presented. The layered bitstream generated by this encoder allows SNR scalability, and moreover, when processed by a JSCD, provides increased robustness to transmission errors compared with a single layered bitstream.**

I. Introduction

To transmit multimedia contents over mixed wired and wireless channels, two main problems have to be solved. First, due to the scarcity of bandwidth on the wireless channels, data generated by multimedia sources have to be efficiently compressed [1]. Second, the compressed bitstreams have to be made robust to transmission impairments, which are unavoidable on wireless channels. Compression standards like JPEG2000 [2], H.264/AVC [3], or Vidwav [4] achieve high compression ratio. They are however very sensitive to transmission errors. Indeed, a single bit error affecting the compressed bitstream may lead to a loss in synchronization at the decoder side, which can significantly affect the quality of the reconstructed multimedia content.

Forward Error Correction (FEC) [5], [6] is the classical approach to increase the robustness of compressed bitstreams. However, when the channel conditions are good, redundancy introduced by FECs may be oversized, and when the channel conditions are bad, residual errors may be present. FECs with adaptive level of protection, see, *e.g.*, [7] are available, but they require a good evaluation of the channel conditions, and thus a feedback channel, which is very difficult to put at work in multi/broadcasting scenarios [8]. Error concealment techniques [9], [10] may be used to mitigate the effect of damaged parts of the transmitted bitstream exploiting the spatial and/or temporal redundancy of the data to decode, but this remains a last-resort solution.

In situations where retransmission of damaged data is difficult, joint source-channel decoding (JSCD) techniques may be very useful, since these techniques exploit residual redundancy left by the source encoder to detect and correct transmission errors remaining after FEC done at lower layers of the protocol stack [11]. Contents of data packets which would be rejected, since they are corrupted by errors, may be processed by JSCD techniques. For that purpose, protocol layers permeable to transmission errors have to be implemented at the receiver side [12], [13], [14].

JSCD techniques may be helped when some side information or redundancy has been deliberately introduced via joint source-channel coding (JSC) in the generated bitstream. This redundancy may take various forms. In some cases, its introduction requires some modifications of the compression standard, as with reversible variable-length codes [15], with the introduction of synchronization markers [16], or error-detecting arithmetic codes [17]. In other situations, opportunistic use of existing features in standards allows to embed side information in packets which will only be processed by JSCD-compliant decoders and will remain transparent to other decoders, see, *e.g.*, additional packets which may be generated in the context of H.264/AVC [3].

JSCD techniques have been successfully applied to several multimedia coders including JPEG2000 [18], MPEG4 [19], H.263+ [20], H.264/AVC [21], or MPEG4/AAC . Preliminary results have been obtained recently [22] for state-of-art DWT-based video coders such as Vidwav [4]. Recent entropy coders such as 3D-ESCOT [23] used in Vidwav leave only very little redundancy in the bitstream. Trellis-based decoding technique such as that used in [24] cannot be applied here, since 3D-ESCOT cannot be easily represented by an automaton (from which the trellis is derived) with a manageable number of states. Thus, a sequential decoding approach [25] exploiting some coarse synchronization markers already present in the bitstream generated by the 3D-ESCOT has been used in [22]. Some improvements of the robustness to transmission errors of a single-layer version of the Vidwav coder have been observed.

The aim of this paper is, first, to introduce (in Section II) a generic JSC architecture, able to provide additional side information, if required, to JSCD tools. The main idea is to consider a *reference* decoder at the encoder side. This decoder is already present in the H.26X architectures [3], but not in DWT-based coders. The purpose of the reference decoder is

to provide some information at encoder side on the *nominal* behavior of the decoder when no noise affects the compressed bitstream. Side information reflecting the nominal behavior (number of bits processed by the entropy decoders, CRCs evaluated on some decoded blocks, as in [26], etc.) that may used to detect and correct transmission errors may then be introduced in the compressed bitstream, in some headers or in some additional packets.

Then, this generic architecture is applied to the Vidwav video coder, described in Section III. Finally, the robustness to transmission errors provided by the JSCD of the layered bitstream generated by the Vidwav coder is illustrated in Section IV. The performance improvements compared to a single-layer JSCD bitstream (see [22]) and to a classically decoded bitstream are illustrated in Section IV.

II. Joint source-channel coder/decoder

Figure 1 sketches a conventional multimedia transmission system. A source encoder delivers a compressed bitstream. Usually, the last operation performed by the source coder is entropy coding using Huffman-like codes or arithmetic codes [1]. The bitstream is packetized to be transmitted over a packet-switched network [27], which is here part of the transmission channel. Each packet consists of a header and a payload containing the encoded data. Packets may be corrupted by noise introduced mainly by the wireless part of the transmission channel.

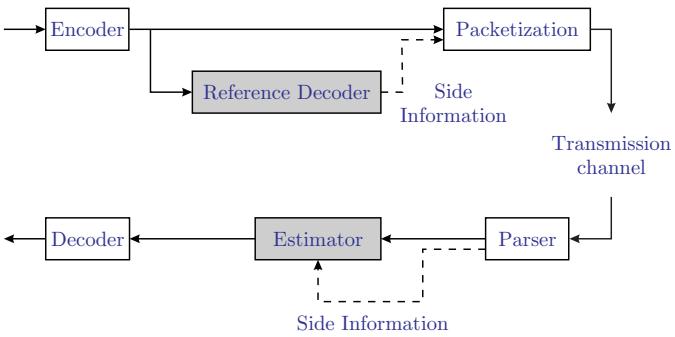

Figure 1: Representation of a multimedia transmission system, with optional joint source-channel coding/decoding devices in gray.

At the receiver side, the content of packets is parsed before source decoding. Parsing may be useful, *e.g.*, to detect packets which are affected by transmission errors after channel decoding using CRCs or checksums at intermediate protocol layers. In this case, conventional transmission systems involve retransmission mechanisms [27], when possible, or use error concealment of lost data [9], [10].

A. Joint source-channel decoder

Before resorting to error concealment, JSCD techniques may be used to correct noisy packets. JSCD is performed at the decoder side, as illustrated in Figure 1. It involves some residual redundancy present (i) in the compressed bitstream and (ii) due to the packetization process: Packet headers may provide very useful information on the content of the payload, and sometimes on the decoded data [20], [11], see also Section III. Only corrupted packets are processed by JSCD, which significantly reduces the computational overhead of these techniques in practical situations.

In what follows, packet headers are assumed to be correctly received. They may have been protected using FECs and/or reliably decoded using joint protocol-channel decoding techniques [14]. Moreover, we assume that the packet payloads consist of *base units* which may be independently decoded.

For each base unit detected as erroneous (when this detection is possible) in a corrupted packet, JSCD techniques involve usually in a maximum-likelihood (ML) or a maximum *a posteriori* (MAP) estimator. The aim is to provide the base unit which is the most likely to have been generated by the source coder. For that purpose, confidence information (soft information) on the bits at the channel output or at the output of channel decoders is used, see Section II-C.

B. Optional joint source-channel coder

JSCD may be helped by some additional side information generated by a reference decoder at the encoder side. To generate such side information, we introduce a *reference* decoder at the encoder side. Some parts of this encoder are already present in classical video coding structures like H.26X video coders [28]. However, the entropy decoder is usually not considered. The aim of the reference decoder, which is exactly a copy of the decoder at receiver side, is to provide some information on the nominal behavior of the decoder at receiver side in absence of transmission errors.

The reference decoder allows thus to get, *e.g.*, the number of processed bits by the entropy decoder for each base unit of encoded data, the number of non-zero coefficients within a temporal or spatial subband, or some checksum evaluated on the pixels of a decoded picture or a part of it, *etc.* This side information may be more or less costly to transmit. In some standards, it may be incorporated in some fields of the headers, as in MPEG4/AAC [29], or in some additional packets as in H.264/AVC [3].

C. Estimation module

Assume that a transmission error has been detected within some base unit $\mathbf{s}$ of N_s bits. One may, for example, determine the MAP estimate $\widehat{\mathbf{s}}_{\text{MAP}}$ of $\mathbf{s}$ from noisy measurements $\mathbf{y}^s$. Here, we assume that $\mathbf{y}^s$ contains *only* noisy versions of the bits in $\mathbf{s}$ and no additional bits from the previous or the next base units. This requires the location of each base unit within a payload to known to the decoder. This information may be extracted from the packet header. It may also be done by robust segmentation of the base units within a payload, see [20].

The MAP estimate has to be compliant with the syntax of the source coder, *i.e.*, $\widehat{\mathbf{s}}_{\text{MAP}}$ should be one of its possible outcome. It should also be compliant with side information introduced during the packetization process with the help of

the reference decoder. Thus

$$\widehat{\mathbf{s}}_{\text{MAP}} = \arg\max_{\mathbf{s}\in\mathcal{C}} p\left(\mathbf{s}|\mathbf{y}^s\right), \tag{1}$$

where $\mathcal{C}$ is the set of all the sequences of N_s bits compliant with the syntax of the source coder and with additional side information. Since all the sequences belonging to $\mathcal{C}$ have the same length of bits, one may assume that they have the same *a priori* probability, which leads to :

$$\begin{aligned}\widehat{\mathbf{s}}_{\text{MAP}} &= \arg\max_{\mathbf{s}\in\mathcal{C}} p\left(\mathbf{y}^s|\mathbf{s}\right), \\ &= \widehat{\mathbf{s}}_{\text{ML}},\end{aligned} \tag{2}$$

the maximum-likelihood estimate. Obtaining the solution for (2) with a reasonable complexity is not trivial in general since the search space $\mathcal{C}$ is usually not well structured. Trellis descriptions of all possible sequences in $\mathcal{C}$ may be obtained when the entropy code is a Huffman-like code [30] or a quasi arithmetic code [31]. For more sophisticated entropy codes, such as context-based adaptive arithmetic codes [32] or 3D-EBCOT, one has to resort to sequential decoding schemes.

D. Sequential decoding scheme

An approximate solution of (2) may be obtained with a manageable complexity with the help of sequential decoding algorithms such as the M-Algorithm (MA) [25]. All sequences of N_s bits may be represented by a binary tree $\mathcal{T}$ of $2^{N_\mathbf{s}}$ leaves. A given node at *depth* n in $\mathcal{T}$ correspond to a partially processed sequence of n bits. This sequence is represented by a *path* of length n in $\mathcal{T}$ stemming from the root of $\mathcal{T}$ and leading to the considered node. Due to the redundancy of the source coder or provided by the side information, only a subset of leaves in $\mathcal{T}$ correspond to sequences belonging to $\mathcal{C}$.

The aim of the MA is to efficiently explore only a part of the tree $\mathcal{T}$. At each iteration, a list $\mathcal{L}$ containing at most M sequences is kept for the next iteration. These sequences are those with the largest metric derived from (2) and given by

$$\mathcal{M}(\mathbf{s}_{1:n}, \mathbf{y}_{1:n}) = -\log p\left(\mathbf{y}_{1:n}|\mathbf{s}_{1:n}\right), \tag{3}$$

where $\mathbf{s}_{1:n}$ denotes the bits indexed from 1 to n of $\mathbf{s}$. Moreover, all partially processed sequences which may be proved not to be prefixes of sequences in $\mathcal{C}$ are discarded.

The steps of the MA are

1. Start with the empty sequence in $\mathcal{L}$ corresponding to the root of $\mathcal{T}$, to which the null metric is assigned.
2. Extend all the paths in $\mathcal{L}$ to the following nodes in $\mathcal{T}$ and calculate their metric.
3. Discard all path which are detected not being prefixes of sequences in $\mathcal{C}$.
4. Keep at most the M paths with the largest metric (3).
5. Go to Step 2 until all paths reach N_s bits or $\mathcal{L} = \emptyset$.

The parameter M of the MA helps to trade-off complexity and decoding efficiency. At the end of the MA, if $\mathcal{L}$ is not empty, it contains only sequences which are have not been detected not to be compliant with the syntax of the source coder and some additional side information[1]. The sequence with the largest metric is then fed to the decoder. When $\mathcal{L}$ is empty at the end of the MA, a path corresponding to the prefix of the actually generated base unit has been discarded due to its too low metric. The MA may be started again with a larger M, increasing then the decoding complexity, but also the probability of finding the solution of (2).

Step 3 requires to detect sequences which are not prefixes of sequences in $\mathcal{C}$. Usually, this operation requires the most computational efforts. To reduce decoding complexity, it may be performed, *e.g.*, only every K iterations of the MA, or at some appropriate iterations.

III. Application to the Vidwav video codec

A. Overview

Vidwav [4] is a 3D wavelet video codec with motion compensated temporal filtering (MCTF), see Figure 2. The

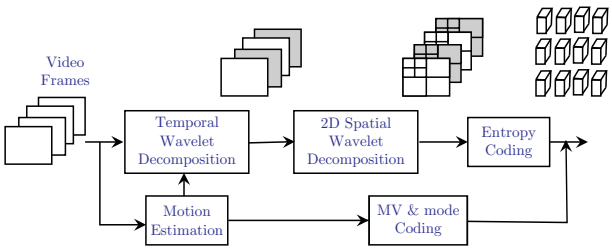

Figure 2: Vidwav coding scheme

subbands generated by the temporal and spatial modules are divided into several 3D *blocks* which are independently entropy coded. The compressed bitstream associated to each 3D block may then be decoded independently from the other bitstreams and are considered are the base units defined in Section II. For each 3D block, a bit-plane coding based on 3D-ESCOT (involving several *coding passes*) is followed by a context-based arithmetic coding.

The 3D-ESCOT entropy coder generates an embedded bitstream for each 3D block. The bitstream is then packetized and the final output compressed sequence is generated. It consists thus of many embedded bitstreams, each of which corresponds to one 3D coding block.

1) Layered bitstream generation: The Vidwav encoder may produce a fully embedded and scalable output by generating a multiple-layer bitstream. To each layer is assigned a given target bit rate. To build the bitstream of a layer at some target bit rate, it is necessary to determine, for each coding block, how many bytes of its embedded bitstream have to be included. This is done with a rate-distortion optimization, determining for each 3D block the number of coding passes that have to be included. This repartition of the bitstream of 3D blocks between layers is illustrated in Figure 3. Here, only three layers are considered.

[1]The double negation is important, since some sequences may not be compliant with the various constraint, but this may not be detectable.

2) Packetisation process: The packetisation process is started once the entropy coding of all 3D blocks is done. It is illustrated in Figure 3. The bitstream of Layer k consists of a header and of several packets $P_{k,i}$, each of which corresponds to one component (Y, U, or V) and to one temporal subband included in Layer k. The i-th packet $P_{k,i}$ of Layer k consists of a packet header $H_{k,i}$ and a packet body $B_{k,i}$. $H_{k,i}$ contains motion information and the set $\mathcal{B}_{k,i}$ of 3D blocks that are included (or eventually partially included in the multi-layer case) in $P_{k,i}$. For each 3D block with index $j \in \mathcal{B}_{k,i}$, $H_{k,i}$ also stores the number of coding passes $p_{k,i,j}$ included in the layer as well as the number of corresponding bytes $n_{k,i,j}$ included in the packet body. The packet body $B_{k,i}$ consists thus of $|\mathcal{B}_{k,i}|$ sequences of bits each one corresponding to a substream extracted from an embedded 3D block.

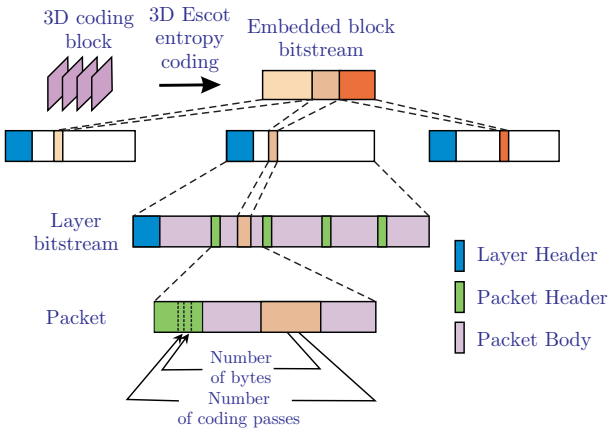

Figure 3: Packetisation Process

B. Robust decoding scheme

This section describes the way the JSCD scheme presented in Section II has been adapted to the Vidwav video codec.

Assume that $K_{\max}$ layers have been generated. Since all headers have been considered to be received without error, it is possible to recover at decoder side for a given 3D block j, the number of coding passes $p_{k,j}$ and the corresponding number of bytes $n_{k,j}$ stored in the bitstream of Layer k, $k = 1, \ldots, K_{\max}$. One may also obtain the observation $\mathbf{y}^s$ of the whole bitstream $\mathbf{s}$ corresponding to the 3D block j, since the number of bytes of each 3D block has been stored in the packet header.

A simple test to verify the compliance of the decoded bitstream with the information provided by the packet headers may then be constructed. This test is based on the fact that an entropy coder may be desynchronized in presence of some corrupted bits, see [33]. The number of bits processed by the decoder may then not be equal to the number of bits generated by the encoder. Thus, if for some candidate estimate $\widetilde{\mathbf{s}}$, decoding of $\sum_{\ell=1}^{k} p_{\ell,j}$ coding passes requires strictly more or strictly less than $\sum_{\ell=1}^{k} n_{\ell,j}$ bytes of the coded bitstream to be processed, then $\widetilde{\mathbf{s}} \neq \mathbf{s}$. This test may be done for each layer $k = 1, \ldots, K_{\max}$.

This test may be used in the MA of Section II to eliminate sequence which do not belong to $\mathcal{C}$. For that purpose, at each iteration of the MA, and for each partially processed sequences, the number of coding passes already decoded has to be kept in memory. Once for some sequence $\sum_{\ell=1}^{k} p_{\ell,j}$ coding passes have been decoded, the number of processed bytes has to be compared to $\sum_{\ell=1}^{k} n_{\ell,j}$ to determine whether the sequence has to be kept or dropped. This test allows to detect errors which desynchronize significantly the entropy decoder.

A more reliable test may be obtained by determining the number of *bits* $\eta_{k,j}$ required to decode $\sum_{\ell=1}^{k} p_{\ell,j}$ coding passes for the j-th 3D block. This information, provided by the reference decoder (see Section II-B), is not stored in the headers but could easily be introduced with a little additional cost. Errors leading to a single bit desynchronisation may the be detected. Nevertheless, in some situations, successive errors may lead to resynchronization, leading to undetectable errors if no other source of redundancy is exploited.

IV. Experimental results

Experiments have been performed using the transmission scheme presented in Section 1. A single layer bitstream has been considered first, before analyzing the performance of the proposed approaches on a three-layer bitstream.

The first 32 frames of `foreman.qcif` have been encoded with a spatial transform and a temporal transform, each one involving 3 levels of decomposition. In the first set of experiments, Vidwav generates a compressed bitstream formed by a single layer encoded at 128 kbps. In the second set of experiments, three layers are generated, corresponding to cumulated bitrates of 32 kbps (1 layer), 64 kbps (2 layers), and 128 kbps (3 layers).

These bitstreams are then sent to an Additive White Gaussian Noise channel (AWGN) with an SNR level going from 10 dB to 13 dB. Data in headers, which represent about 18% of the bitstream, are assumed to be received without error. All results have been averaged over 100 noise realizations.

A reference decoder is used at encoder side, as indicated in Section III-B to determine the nominal behavior of the entropy decoder in absence of transmission errors. For each 3D block, and for each layer (in the multi-layer case), the number of processed bits is written as side information. This information is already partly present in each packet header. In average, only three additional bits have to be supplemented for each encoded block. The amount of redundancy introduced in the bitstream is about 0.40% in the case of a single-layer bitstream and about 1.3% in the three-layer case. In both cases, this redundancy is negligible compared to the redundancy that could be introduced by any channel code. The amount of introduced redundancy is larger in the three-layer case, since the compressed stream for each block is partitionned into various layers, for each of which side information has to be provided.

A. One layer

Table I compares the proposed JSCD schemes (with and without side information) and a standard decoder. The gain in PSNR observed with the informed JSCD is always more important than the one obtained with the non-informed JSCD, when compared to the standard decoder.

Table II illustrates the evolution of the proportion of erroneous blocks and of blocks detected as erroneous (BDE) as a function of the channel SNR for different values of M and for various decoders. When a block is detected as erroneous, it is actually corrupted by transmission errors. Some errors, however may not be detected. The results with $M = 1$ indicate the proportion of BDE without having performed any correction. Those with $M = 10$ indicate the residual proportion of BDE after correction, and represent thus a lower bound on the proportion of blocks still erroneous in the decoded stream.

Channel SNR	10	10.5	11	11.5	12
SD	19.0449	22.0745	25.0485	29.3347	31.2268
JD RR $M = 2$	19.5972	22.9858	27.4965	31.7263	32.9001
JD RSI $M = 2$	19.7546	23.0609	27.5170	31.7271	32.9029
JD RR $M = 6$	20.3220	25.9533	30.7641	33.5468	33.4892
JD RSI $M = 6$	20.3436	26.0417	30.8123	33.5491	33.4989
JD RR $M = 10$	20.3236	26.8208	31.9860	33.8753	33.8940
JD RSI $M = 10$	20.4424	26.8531	32.0196	33.8857	33.9015

Table I: Average PSNR of the standard decoder (SD) and the proposed joint decoders (JD) when only the residual redundancy (RR) is used and when some side information (RSI) is used in addition, for $M = 2, 6$ and 10 and for a single-layer bitstream.

When only residual source redundancy is used, 89.97% of erroneous blocks are detected in average. This proportion increases to 94.60% when the side information is used as well.

When $M = 10$, side information still allows more erroneous blocks to be detected compared to a JSCD without side information. One may increase M to reduce the amount of corrected blocks, the price to be paid is an increased complexity.

Channel SNR	10	10.5	11	11.5	12
EB	22.3189	13.0720	7.0906	3.2618	1.5153
BDE RR $M = 1$	20.7214	12.1045	6.3259	2.9499	1.2841
BDE RR $M = 10$	2.6727	0.7911	0.3050	0.1783	0.1295
BDE RSI $M = 1$	21.7772	12.6908	6.6518	3.1086	1.3524
BDE RSI $M = 10$	3.0669	0.8914	0.3468	0.2173	0.1504

Table II: Percentage of erroneous blocks (EB) and blocks detected as erroneous (BDE) as a function of the SNR for the robust decoder using only the residual redundancy (RR) and for the JSCD scheme exploiting both the residual redundancy and the side information (RSI), for a single layer bitstream.

B. Three layers

In this part, the output bitstream is organized in three layers. Side information is generated for each layer. Figure 4 compares the average PSNR obtained at the output of a JSCD and at the output of a standard decoder using the side information to detect whether a block is erroneous (informed standard decoder). This decoder performs more efficiently than a classical (non-informed) decoder, since it is able to remove blocks deemed as erroneous and belonging to refinement layers. The decoding of one, two, and three layers has been performed.

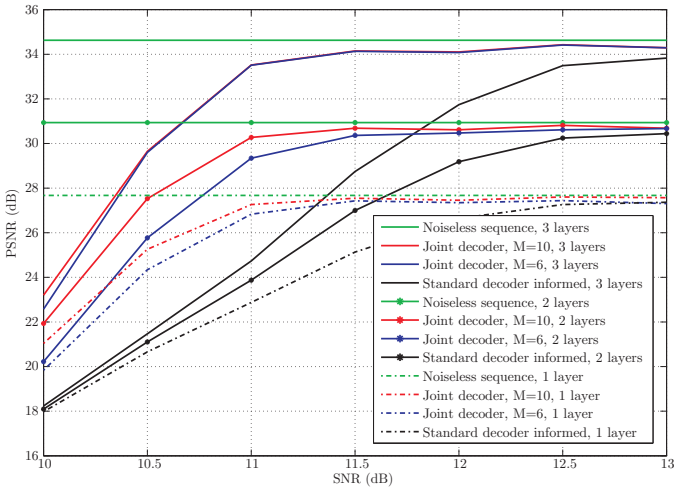

Figure 4: Average PSNR of the decoded sequence produced by an informed standard decoder and by the proposed JSCD as a function the channel SNR when 1, 2, and 3 layers are decoded.

When $M = 10$, and when all layers are decoded, the gain in PSNR is more than 8.5 dB at an SNR of 11 dB, when one compares the JSCD with the standard informed decoder. In terms of SNR, the gain is about 1.2 dB. Using $M = 10$ instead of $M = 6$ in the JSCD provides some improvement in terms of PSNR, especially at low SNRs, mainly for Layer 1 and Layer 2. The improvement becomes negligible for Layer 3. This is mainly due to the fact that blocks which are erroneous in Layers 1 and 2 and were not detected as erroneous do not allow blocks at Layer 3 to be corrected. Even blocks at Layer 3 which were error-free may be deemed as erroneous. Thus, the JSCD scheme is less efficient for higher (low-priority) layers.

Table I and Figure 4 allow also to compare the results provided by the JSCD on the single-layer bitstream and those for the multiple-layer bitstream. One sees that at equivalent bitrates, the decoding schemes working on three-layer bitstreams provide improved PSNRs. The gain is about 3 dB at an SNR of 10 dB and 1.5 dB at an SNR of 11 dB.

Figure 5 shows the qualitative improvement brought by the proposed technique on a frame of the decoded sequence for the 3-layered bitstream when compared to the standard informed decoder.

Figure 5: Frame 16 of `foreman.qcif` sent over an AWGN with SNR = 11 dB decoded with the standard decoder *(left)* and with the proposed robust decoder using the side information *(right)*, M=10.

V. Conclusions

This paper introduces a JSCD scheme able to detect and correct many transmission errors affecting a Vidwav encoded scalable bitstream. This decoder uses soft information on the bits to decode, combined with residual source redundancy and information linked to the packetization of compressed data.

Additional information provided by a reference decoder at the encoder, providing the nominal bahavior of the decoder in absence of transmission errors is also very useful. In this paper, the reference decoder introduces some information on the bahavior of the entropy decoder, which leads to a redundancy of about 1% of the length of the original compressed bitstream.

Significant performance improvements of more than 8.5 dB in PSNR and 1.2 dB in SNR have been observed thanks to the proposed JSCD scheme. Experiments have also illustrated that considering several layers provides an improved robustness compared to the single-layer scheme, even if combined with JSCD. Moreover, this JSCD techniques introduces a reasonable computationnal overhead, especially at high channel SNR, since only packets with error are robustly decoded.

An optimization of the decoding effort depending on the considered layer may probably be helpful, since when errors are remaining at lower layers, upper layers are lost, even if they are error-free. Other source of redundancies introduced by the reference decoder may also be investigated. Finally, other transmission channels have to be considered.

References

[1] K. Sayood, *Introduction to Data Compression, Second Edition*. San Francisco: Morgan Kaufmann, 2000.

[2] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG-2000 still image compression standard," *IEEE Signal Proc. Mag.*, pp. 36–58, 2001.

[3] ITU-T and ISO/IEC JTC 1, "Advanced video coding for generic audiovisual services," ITU-T Rec. H.264, and ISO/IEC 14496-10 AVC, Tech. Rep., nov. 2003.

[4] R. Xiong, X. Ji, D. Zhang, J. Xu, G. Pau, M. Trocan, S. Brangoulo, and V. Bottreau, "Vidwav wavelet video coding specifications," MPEG document, Tech. Rep., Poznan, July 2005.

[5] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and applications*. Englewood Cliffs: Prentice-Hall, 1983.

[6] T. Richardson and U. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008.

[7] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Communications*, vol. 36, no. 4, pp. 389–400, 1988.

[8] ETSI, "Digital video broadcasting (DVB); transmission system for handheld terminals (DVB-h)," ETSI EN 302 304 v1.1.1, Tech. Rep., nov. 2004.

[9] Y. Wang and Q. Zhu, "Error control and concealment for video communication: A review," *Proc. of the IEEE*, vol. 86, pp. 974–997, 1998.

[10] M. C. Hong, H. Schwab, L. P. Kondi, and A. K. Katsaggelos, "Error concealment algorithms for compressed video," *Signal Processing: Image Communication*, vol. 14, pp. 473–492, 1999.

[11] P. Duhamel and M. Kieffer, *Joint source-channel decoding: A cross-layer perspective with applications in video broadcasting*. Academic Press, 2009.

[12] G. Panza, E. Balatti, G. Vavassori, C. Lamy-Bergot, and F. Sidoti, "Supporting network transparency in 4G networks." in *Proc. IST Mobile and Wireless Communication Summit*, 2005.

[13] H. Jenkac, T. Stockhammer, and W. Xu, "Permeable-layer receiver for reliable multicast transmission in wireless systems," in *Proc. IEEE Wireless Communications and Networking Conference*, vol. 3, 13-17 March 2005, pp. 1805–1811.

[14] C. Marin, Y. Leprovost, M. Kieffer, and P. Duhamel, "Robust mac-lite and soft header recovery for packetized multimedia transmission," *IEEE Trans. on Communications*, 2010, to appear.

[15] Y. Takishima, M. Wada, and H. Murakami, "Reversible variable length codes," *IEEE Trans. Comm.*, vol. 43, no. 2-4, pp. 158–162, 1995.

[16] Z. W. A. Bilgin and M. Marcellin, "Decompression of corrupt jpeg2000 codestreams," in *Data Compression Conference*, 2003.

[17] D. Bi, W. Hoffman, and K. Sayood, "State machine interpretation of arithmetic codes for joint source and channel coding," *Proc. of DCC, Snowbird, Utah, USA.*, pp. 143–152, 2006.

[18] M. Grangetto, E. Magli, and G. Olmo, "Reliable JPEG 2000 wireless imaging by means of error-correcting MQ coder," in *Proc. ICME*, vol. 1, pp. 9–12, 2004.

[19] A. Kopansky and M. Bystrom, "Sequential decoding of MPEG-4 coded bitstreams for error resilience," in *Proc. 33rd Annual Conference on Information Sciences and Systems*, 1999.

[20] C. Lee, M. Kieffer, and P. Duhamel, "Soft decoding of VLC encoded data for robust transmission of packetized video," in *Proceedings of ICASSP*, 2005, pp. 737–740.

[21] G. Sabeva, S. Ben-Jamaa, M. . Kieffer, and P. Duhamel, "Robust decoding of h.264 encoded video transmitted over wireless channels," in *Proceedings of MMSP*, Victoria, Canada, 2006.

[22] M. Abid, M. Kieffer, M. Cagnazzo, and B. Pesquet-Popescu, "Robust decoding of a 3D-ESCOT bitstream transmitted over noisy channels," in *Proc. IEEE Int. Conf. on Image Proc.*, 2010, submitted.

[23] S. L. J. Xu, Z.Xiong and Y. Zhang, *3-D embedded subband coding with optimal truncation (3-D ESCOT)*. J. Appl. Comput Harmon Analysis, vol. 10, pp.290-315, May 2001.

[24] S. Malinowski, H. Jegou, and C. Guillemot, "Error recovery properties and soft decoding of quasi-arithmetic codes," *EURASIP Journal on Advances in Signal Processing*, 2008, to appear.

[25] J. B. Anderson and S. Mohan, *Source and Channel Coding: An Algorithmic Approach*. Kluwer, 1991.

[26] R. Puri and K. Ramchandran, "PRISM: A video coding paradigm based on motion-compensated prediction at the decoder," *IEEE Transactions on Image Processing*, 2005, submitted.

[27] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*, 3rd ed. Boston: Addison Wesley, 2005.

[28] L. Hanzo, P. Cherriman, and J. Streit, *Video Compression and Communications: From Basics to H.261, H.263, H.264, Mpeg4 for DVB and HSDPA-Style Adaptive Turbo-Transceivers*, 2nd ed. John Wiley & Sons Ltd, 2007.

[29] ISO/IEC, "MPEG-4 advanced audio coding, AAC," International Organization for Standardization, Tech. Rep. 14496-3, 2005.

[30] V. Buttigieg and P. Farrell, "Variable-length error-correcting codes," *IEE Proc. on Com.*, vol. 147, no. 4, pp. 211–215, 2000.

[31] T. Guionnet and C. Guillemot, "Soft and joint source-channel decoding of quasi-arithmetic codes," *EURASIP journal on Applied Signal Processing*, vol. 2004, no. 3, pp. 393–411, March 2004.

[32] D. Marpe, H. Schwarz, and T. Weigand, "Context based adaptative binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13(7), pp. 620–636, 2003.

[33] S. Malinowski, H. Jegou, and C. Guillemot, "Synchronization recovery and state model reduction for soft decoding of variable length codes," *IEEE Trans. on Information Theory*, vol. 53, no. 1, pp. 368–377, Jan. 2007.